UNIP

UNIVERSIDADE PAULISTA

Paradigmas de Linguagens

Aula 5: Funções

Professora Sheila Cáceres

Introdução

- Funções são elementos fundamentais em toda linguagem de programação, já que são ferramentas essenciais para modularização e abstração em programação.
- Em linguagens diferentes, as funções são conhecidas como procedimentos, sub-rotinas, subprogramas ou métodos, e possuem diversas características em comum, assim como algumas diferenças importantes nas mais variadas linguagens.

Funções

- **O que são “Funções”?**

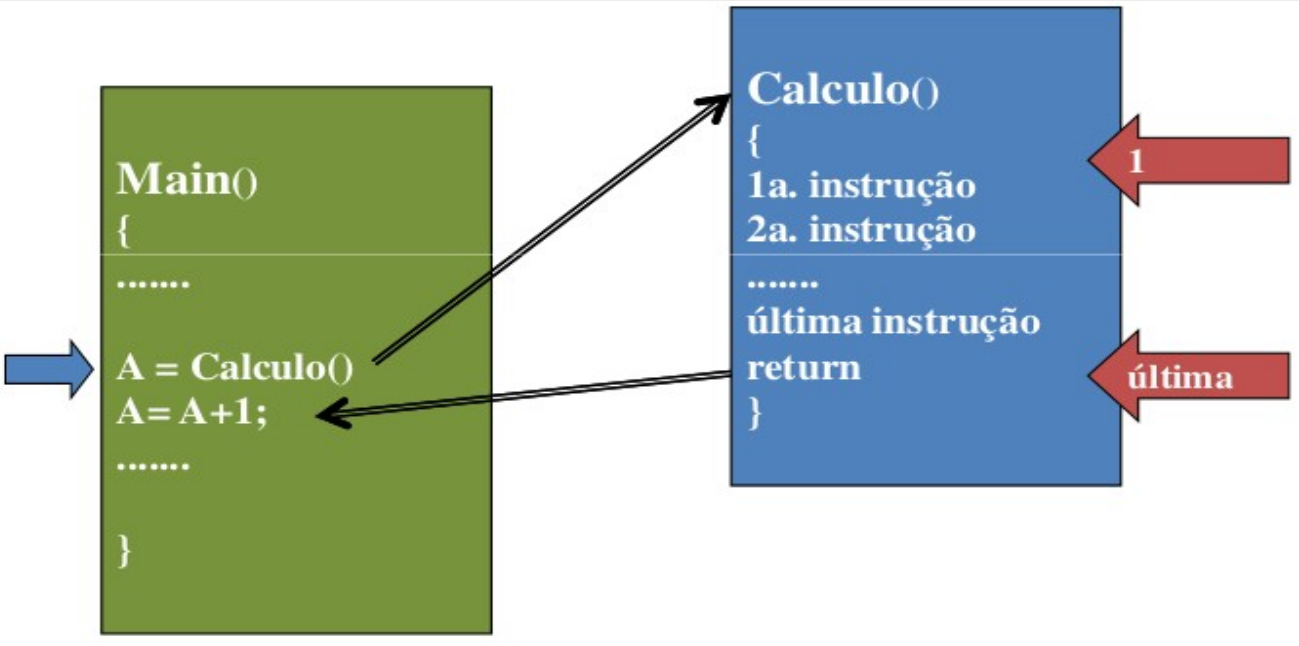
  São trechos de código fonte agrupados sob um nome, que podem ser chamados sempre que for necessário executar uma determinada ação programada neste trecho.

- **Como usar funções?**

  Atribui-se um nome à uma sequência de comandos, e faz-se referência a este nome nos vários lugares do programa onde a sequência em questão deveria ser repetida.

Características

- Cada subprograma tem um único ponto de entrada.
- A unidade de programa chamadora é suspensa durante a execução do subprograma chamado, implicando a existência de apenas um subprograma em execução em qualquer momento no tempo.
- O controle sempre retorna para o chamador quando a execução do subprograma termina.

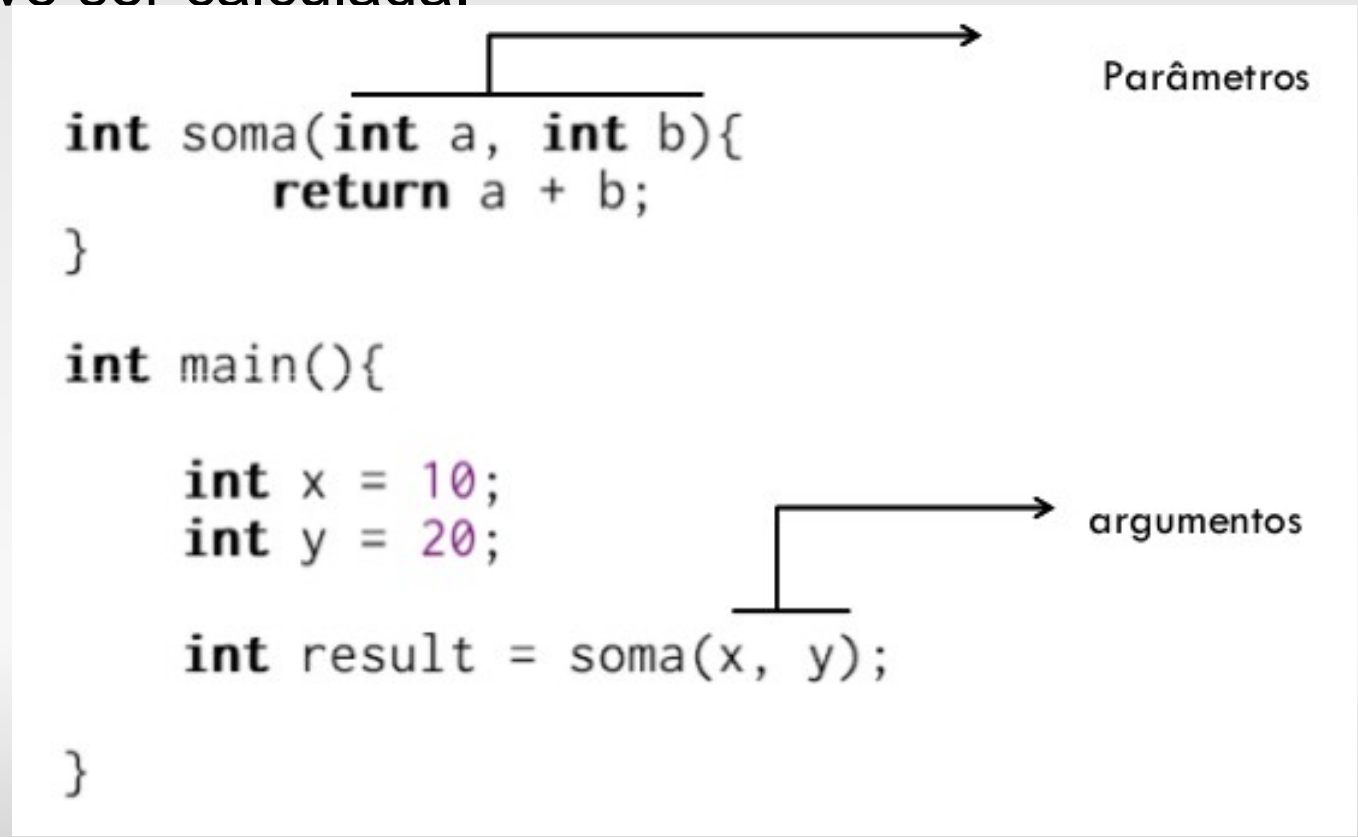


Por que usar funções?

- **Evita** escrita **repetida** de código.
  - Economiza o tempo gasto com o trabalho de copiar estas sequências;
  - Evitar a necessidade de mudar em múltiplos lugares caso deseje alterar o seu funcionamento;
- Dividir grandes tarefas de computação em tarefas menores (**Modularização**):
  - Facilita o gerenciamento de grandes sistemas
  - Aumenta a confiabilidade dos mesmos.

Parâmetros e Argumentos

- **Parâmetro:** Identificador que aparece na declaração de função.
- **Argumento:** Expressão que aparece em uma chamada de função.
- Funções derivam sua grande utilidade pela sua capacidade de receber parâmetros. Ex: Uma função de raiz quadrada seria inútil se não pudesse receber um argumento, que especificasse o valor cuja raiz deve ser calculada.



Mecanismos de passagem de parâmetros

- Os mecanismos mais usados de passagem de parâmetros são:
  - Passagem por valor
  - Passagem por referência

Passagem por Valor

- Passar um argumento por valor significa que o valor do argumento é calculado no tempo da chamada e **copiado** para o parâmetro correspondente.

```
static int soma(int a, int b){
    return a+b;
}

public static void main(String[] args){
    int x=10;
    int y=20;
    int result = soma(x,y);
    System.out.println("A soma é: "+result);
}
```

O resultado impresso na tela é:
A soma é 30

Passagem de parâmetros por Referencia

- Passar um argumento por referência (ou por **endereço**) significa que o endereço de memória do argumento é copiado para o parâmetro correspondente, de modo que o parâmetro se torna uma referência (ponteiro) indireta ao argumento real.
- Assim, todas as atribuições ao parâmetro formal dentro da vida da chamada afetam diretamente o valor do argumento.

Passagem de parâmetros por Referência (Exemplo)

Exemplo na linguagem C

- Dentro do main, a função troca é chamada enviando os endereços das variáveis a e b. Assim, dentro da função troca o valor que tinha a primeira variável é trocada pelo valor existente na segunda variável de forma permanente.
- Após a chamada à função o valor de a será 5 e o de b será 0. Assim, a impressão (printf) dentro do main será 5 0

```
void troca(int *x, int *y)
{
 int aux;
 aux=*x;
 *x=*y;
 *y=aux;
}
```

```
int main()
int main()
{
 int a=0,b=5;
 troca(&a,&b);
 printf("%d %d",a,b);
}
```

Funções Recursivas

- São funções que chamam elas mesmas no corpo das funções.
- São implementadas utilizando-se uma pilha que registra o status de todas as chamadas ativas.

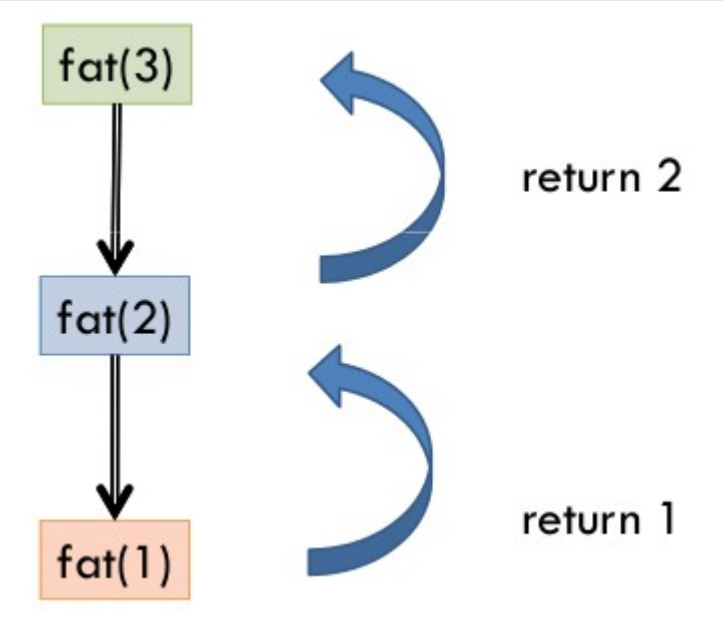
Funções Recursivas

Exemplo em Java:

- Vemos que a função fat se chama a si mesma até chegar a n=1, e com esse resultado consegue finalizar as funções que foram chamadas previamente.

```
static int fat(int n){
    if(n<2){
        return 1;
    }
    else{
        return n*fat(n-1);
    }
}
public static void main(String[] args){
    System.out.println("O fatorial de 3 é: " + fat(5));
}
```

- Se chamar fat(3)



Referências Bibliográficas

O material para a realização desta apresentação foram extraídos de:

* Tucker, Linguagens de Programação, principios e paradigmas.

* Sebesta, Linguagens de programação, 4ta edição.

* VAREJÃO, Flavio. Linguagens de Programação, 2004.